DATA ANALYSIS ON TOP YOUTUBERS

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

In [2]:
```
df = pd.read_csv("/kaggle/input/youtubers-df-csv/youtubers_df.csv")
```

In [3]:
```
df.shape
```

Out[3]:
```
(1000, 9)
```

In [4]:
```
df.head()
```

Out[4]:

	Rank	Username	Categories	Suscribers	Country	Visits	Likes	Comments
0	1	tseries	Música y baile	249500000.0	India	86200.0	2700.0	78.0
1	2	MrBeast	Videojuegos, Humor	183500000.0	Estados Unidos	117400000.0	5300000.0	18500.0
2	3	CoComelon	Educación	165500000.0	Unknown	7000000.0	24700.0	0.0
3	4	SETIndia	NaN	162600000.0	India	15600.0	166.0	9.0
4	5	KidsDianaShow	Animación, Juguetes	113500000.0	Unknown	3900000.0	12400.0	0.0

DATA CLEANING AND DATA EXPLORATION

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Rank        1000 non-null   int64  
 1   Username    1000 non-null   object 
 2   Categories  694 non-null    object 
 3   Suscribers  1000 non-null   float64
 4   Country     1000 non-null   object 
 5   Visits      1000 non-null   float64
 6   Likes       1000 non-null   float64
 7   Comments    1000 non-null   float64
 8   Links       1000 non-null   object 
dtypes: float64(4), int64(1), object(4)
memory usage: 70.4+ KB
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['Rank', 'Username', 'Categories', 'Suscribers', 'Country', 'Visits',
       'Likes', 'Comments', 'Links'],
      dtype='object')
```

In [7]:
```
df.rename(columns = {'Suscribers' : 'Subscribers'}, inplace = True)
```

In [8]:
```
df.columns
```

Out[8]:
```
Index(['Rank', 'Username', 'Categories', 'Subscribers', 'Country', 'Visits',
       'Likes', 'Comments', 'Links'],
      dtype='object')
```

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
Rank            0
Username        0
Categories    306
Subscribers     0
Country         0
Visits          0
Likes           0
Comments        0
Links           0
dtype: int64
```

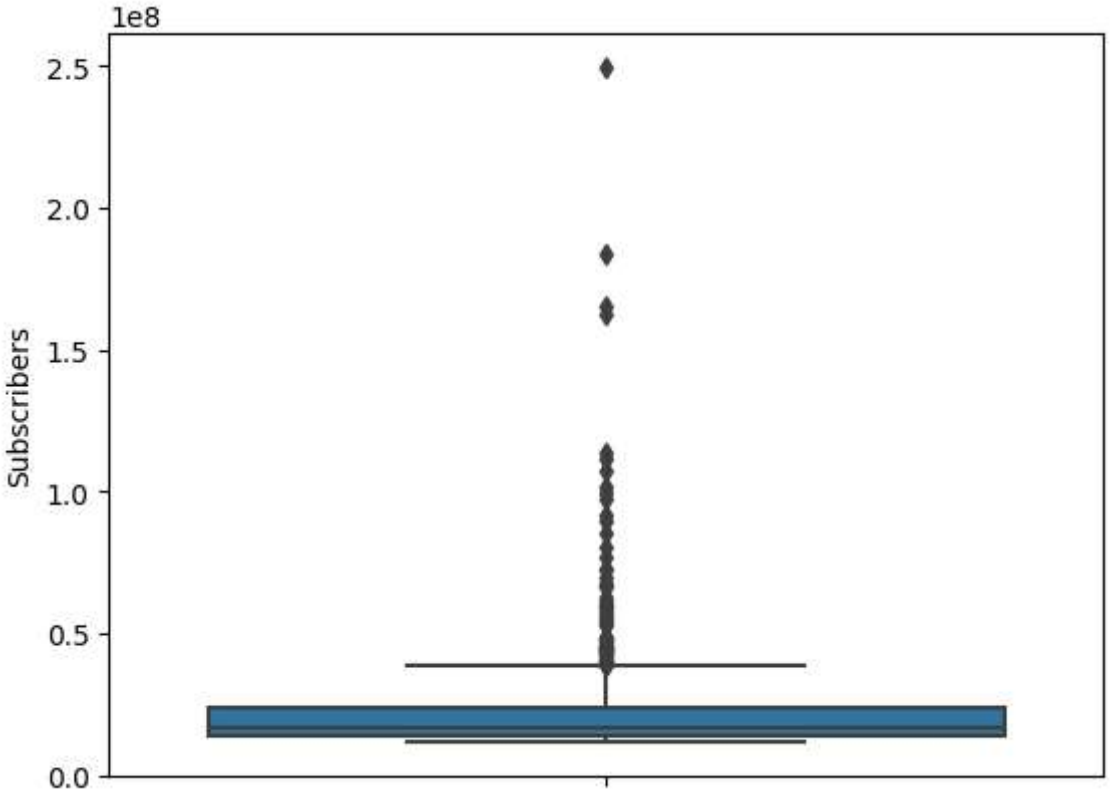
In [11]:
```
df['Categories'].fillna('Unknown', inplace = True)
```

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
Rank           0
Username       0
Categories     0
Subscribers    0
Country        0
Visits         0
Likes          0
Comments       0
Links          0
dtype: int64
```

Check for outliers

In [13]:
```
sns.boxplot(y=df['Subscribers'])
plt.show()
```



REMOVING OUTLIERS

In [14]:
```
#Removing for subscribers
col1 = 'Subscribers'
# Calculate the interquartile range (IQR)
Q1 = df[col1].quantile(0.25)
Q3 = df[col1].quantile(0.75)
IQR = Q3 - Q1
```

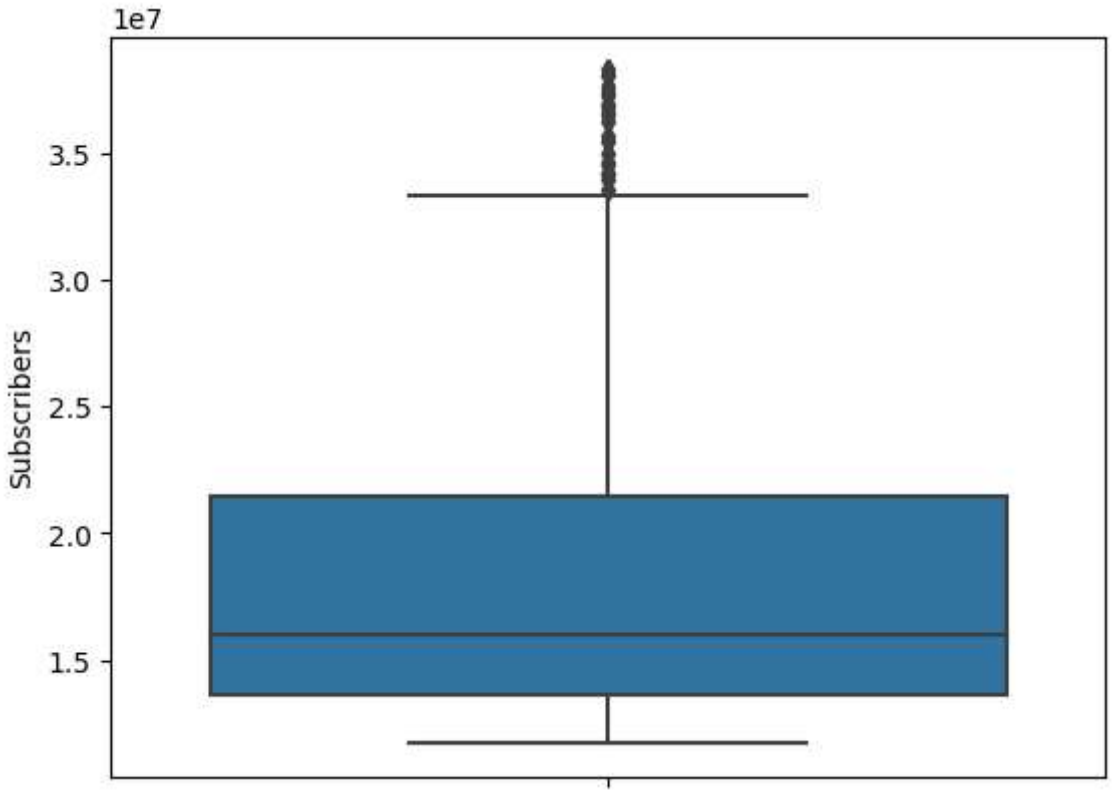
In [15]:
```
IQR
```

Out[15]: 9900000.0

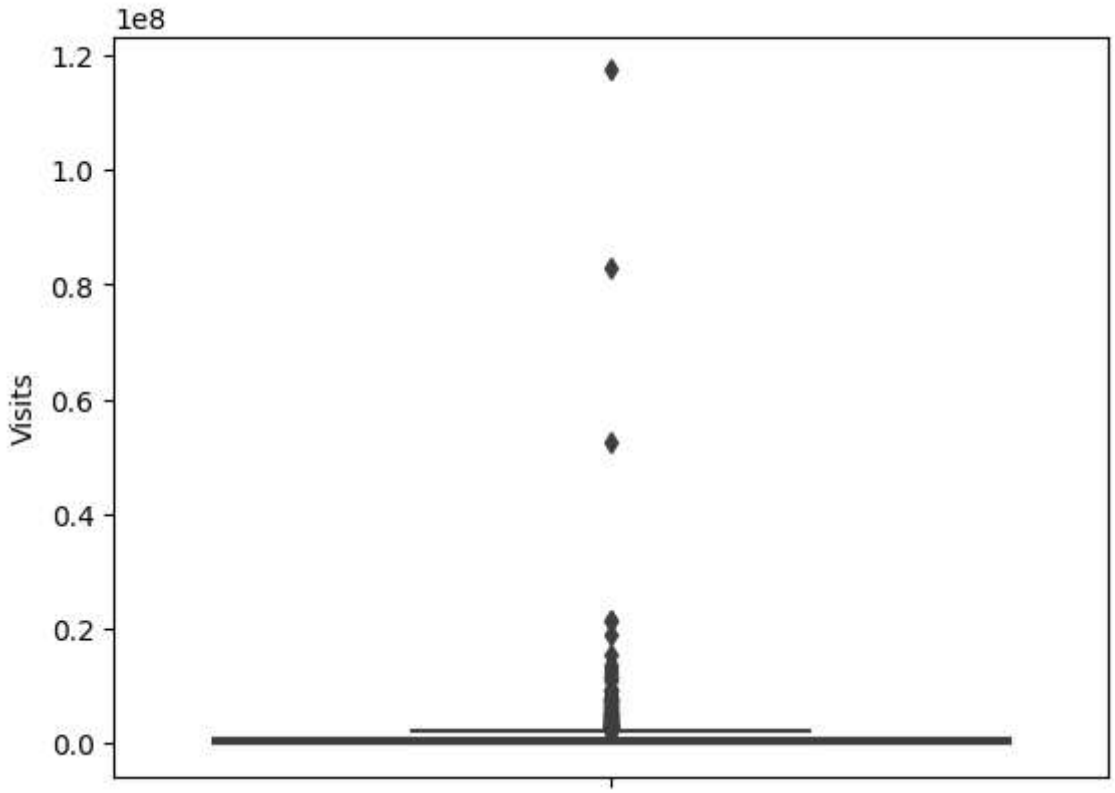
In [16]:
```
# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
```

In [17]:
```
# Filter the dataframe to remove outliers
df_no_outliers = df[(df[col1] >= lower_bound) & (df[col1] <= upper_bound)]
```

In [18]:
```
# Plot the boxplot without outliers
sns.boxplot(y=df_no_outliers[col1])
plt.show()
```



In [19]:
```
sns.boxplot(y=df['Visits'])
plt.show()
```



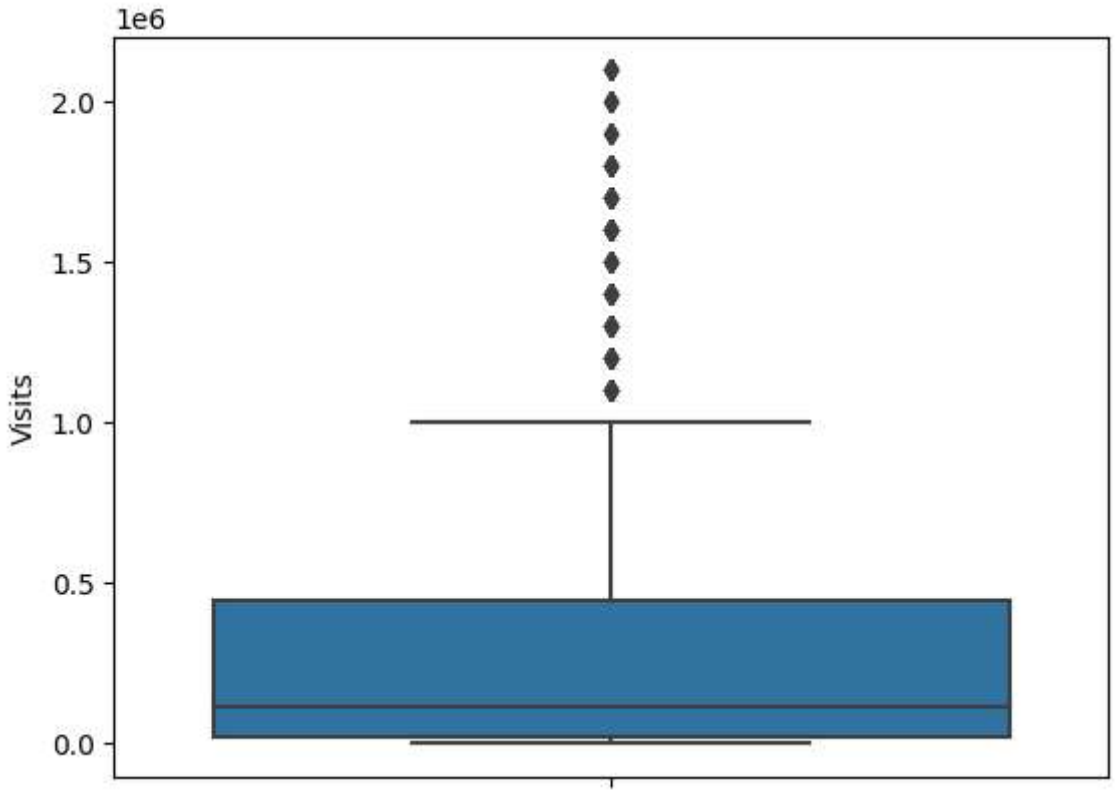
In [20]:
```
#Removing outliers for Visits
col2 = 'Visits'
# Calculate the interquartile range (IQR)
Q1 = df[col2].quantile(0.25)
Q3 = df[col2].quantile(0.75)
IQR1 = Q3 - Q1
IQR1
```

Out[20]: 833500.0

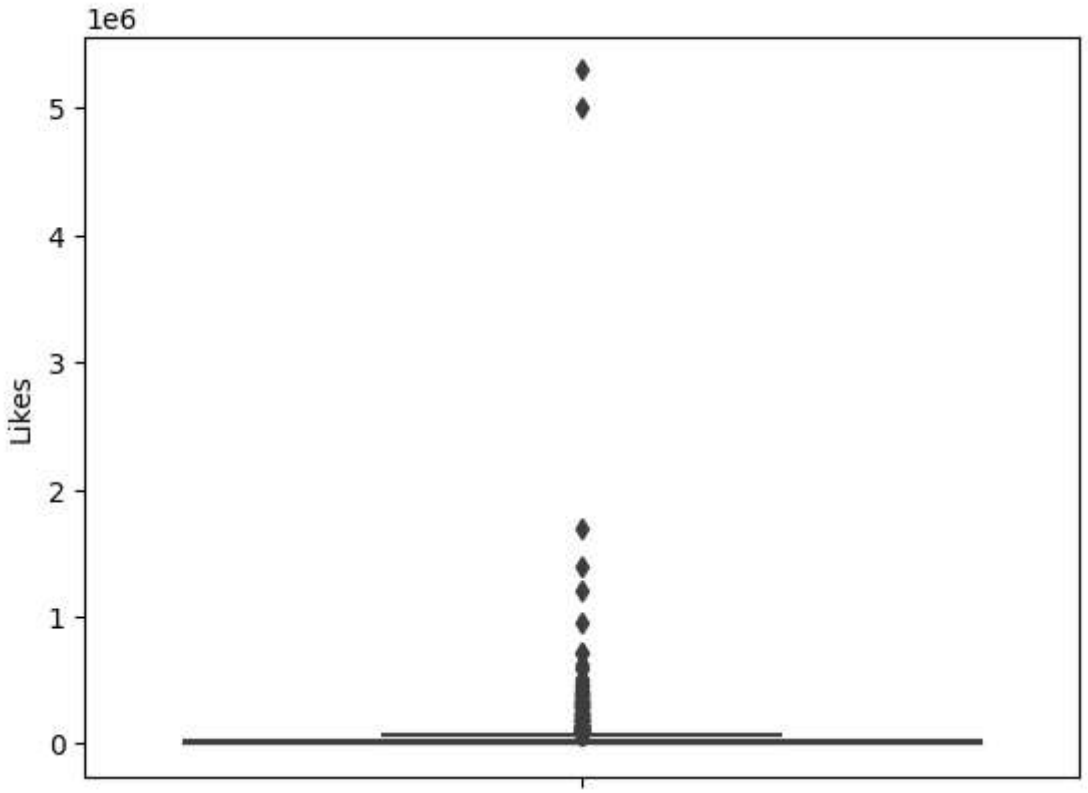
In [26]:
```
# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR1
upper_bound = Q3 + 1.5 * IQR1
```

In [27]:
```
# Filter the dataframe to remove outliers
df_no_outliers = df[(df[col2] >= lower_bound) & (df[col2] <= upper_bound)]
```

In [28]:
```
# Plot the boxplot without outliers
sns.boxplot(y=df_no_outliers[col2])
plt.show()
```



In [30]:
```
sns.boxplot(y=df['Likes'])
plt.show()
```



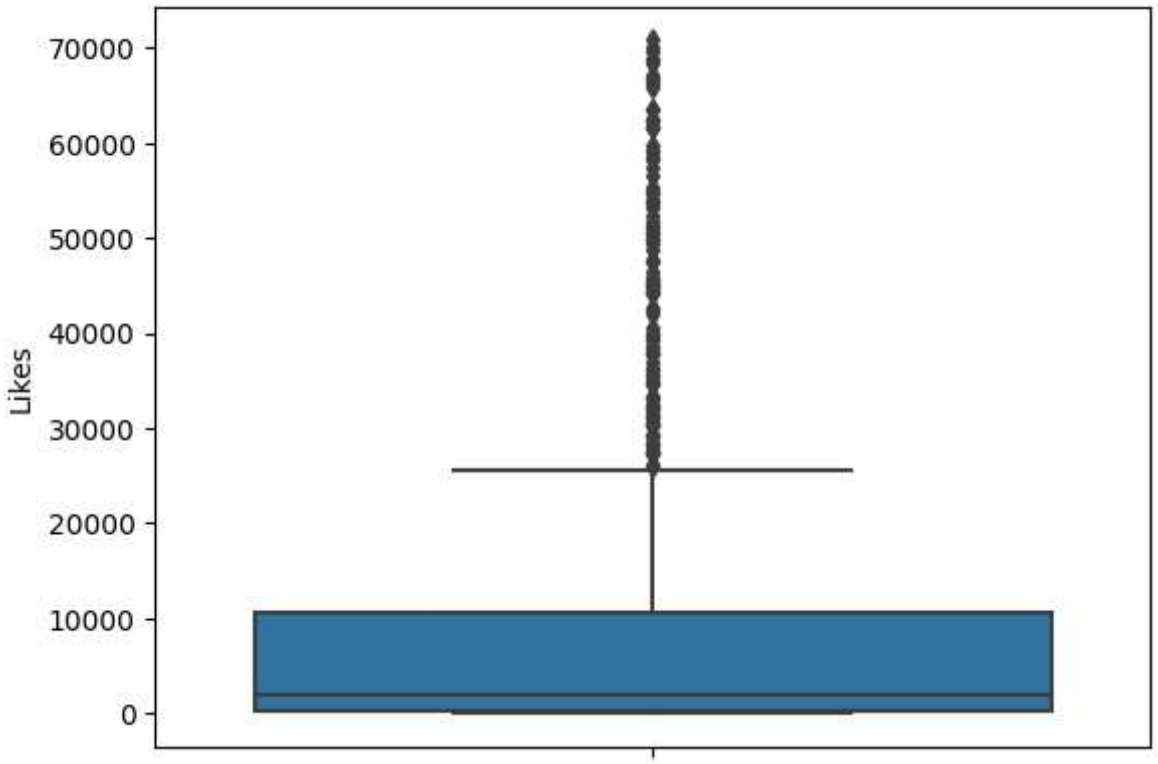
In [31]:
```
#Removing outliers for Visits
col3 = 'Likes'
# Calculate the interquartile range (IQR)
Q1 = df[col3].quantile(0.25)
Q3 = df[col3].quantile(0.75)
IQR2 = Q3 - Q1
IQR2
```

Out[31]: 28178.25

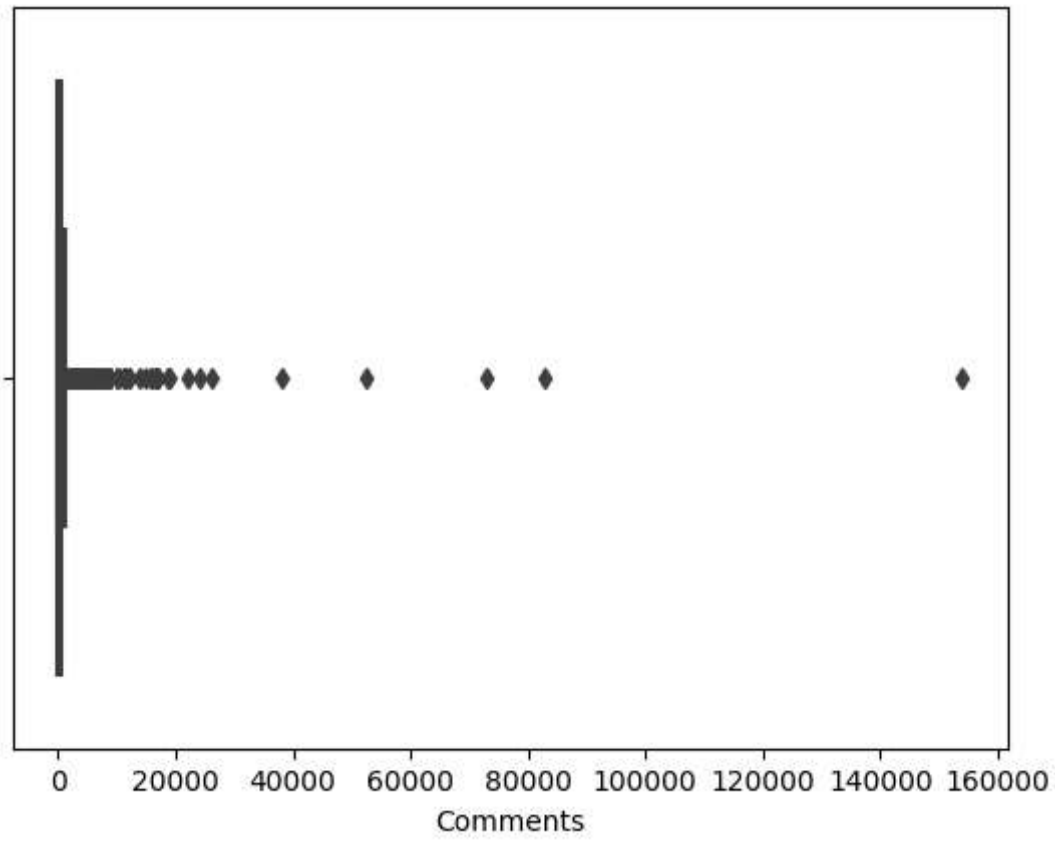
In [32]:
```
# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR2
upper_bound = Q3 + 1.5 * IQR2
```

In [33]:
```
# Filter the dataframe to remove outliers
df_no_outliers = df[(df[col3] >= lower_bound) & (df[col3] <= upper_bound)]
```

In [34]:
```
# Plot the boxplot without outliers
sns.boxplot(y=df_no_outliers[col3])
plt.show()
```



In [35]:
```
sns.boxplot(x=df['Comments'])
plt.show()
```



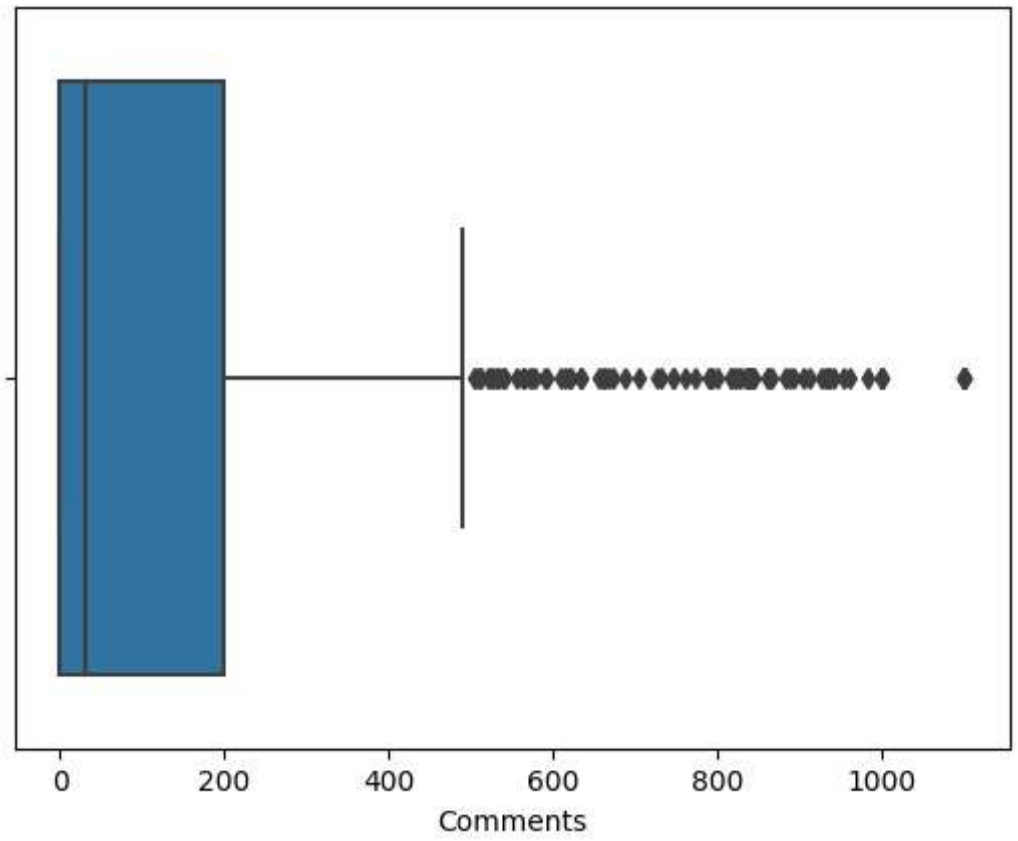
In [36]:
```
#Removing outliers for Visits
col4 = 'Comments'
# Calculate the interquartile range (IQR)
Q1 = df[col4].quantile(0.25)
Q3 = df[col4].quantile(0.75)
IQR3 = Q3 - Q1
IQR3
```

Out[36]: 470.0

In [38]:
```
# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR3
upper_bound = Q3 + 1.5 * IQR3
```

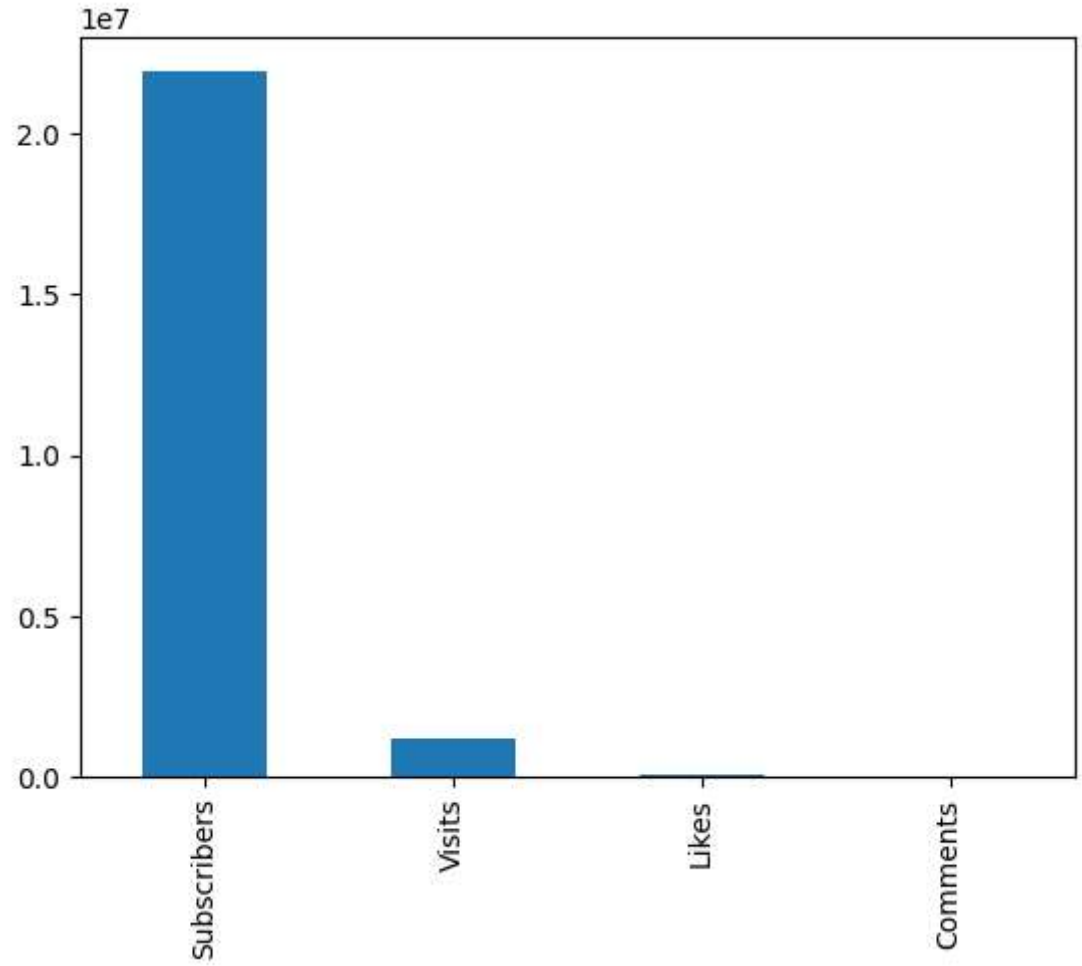
In [39]:
```
# Filter the dataframe to remove outliers
df_no_outliers = df[(df[col4] >= lower_bound) & (df[col4] <= upper_bound)]
```

In [40]:
```
# Plot the boxplot without outliers
sns.boxplot(x=df_no_outliers[col4])
plt.show()
```



Performance Metrics

In [41]:
```
average_metrics = df[['Subscribers', 'Visits', 'Likes', 'Comments']].mean()
average_metrics.plot(kind='bar')
plt.show()
```



Content Categories

In [42]:
```
category_distribution = df['Categories'].value_counts()
print(category_distribution)
```

```
Categories
Unknown                                  306
Música y baile                           160
Películas, Animación                      61
Música y baile, Películas                 41
Vlogs diarios                             37
Noticias y Política                       36
Películas, Humor                          34
Animación, Videojuegos                    34
Animación, Juguetes                       29
Animación, Humor                          27
Películas                                 24
Educación                                 24
Animación                                 22
Videojuegos                               19
Videojuegos, Humor                        17
Música y baile, Animación                 16
Ciencia y tecnología                      14
Comida y bebida                           12
Humor                                     10
Juguetes                                  10
Películas, Juguetes                        9
Películas, Videojuegos                     8
Deportes                                   8
Música y baile, Humor                      6
Juguetes, Coches y vehículos               4
DIY y Life Hacks                           3
Fitness, Salud y autoayuda                 3
Videojuegos, Juguetes                      3
Animales y mascotas                        2
Moda                                       2
Coches y vehículos                         2
Educación, Juguetes                        2
Fitness                                    2
Comida y bebida, Juguetes                  1
ASMR, Comida y bebida                      1
Animación, Humor, Juguetes                 1
Diseño/arte, Belleza                       1
Belleza, Moda                              1
ASMR                                       1
Música y baile, Juguetes                   1
Diseño/arte, DIY y Life Hacks              1
DIY y Life Hacks, Juguetes                 1
Diseño/arte                                1
Comida y bebida, Salud y autoayuda         1
Viajes, Espectáculos                       1
Juguetes, DIY y Life Hacks                 1
Name: count, dtype: int64
```

Performing kmeans clustering to check high performance

In [43]:
```
#This ensures that the values have a mean of 0 and standard deviation of 1
scaler = StandardScaler()
scaled_metrics = scaler.fit_transform(df[['Subscribers', 'Visits', 'Likes', 'C
kmeans = KMeans(n_clusters=2)
df['cluster'] = kmeans.fit_predict(scaled_metrics)
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:870: Futu
reWarning: The default value of `n_init` will change from 10 to 'auto' in 1.
4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

In [45]:
```
# Identify top-performing content creators
top_performers = df[df['cluster'] == 1]
```

In [46]:
```
top_performers
```

Out[46]:

	Rank	Username	Categories	Subscribers	Country	Visits	Likes	Comments	
1	2	MrBeast	Videojuegos, Humor	183500000.0	Estados Unidos	117400000.0	5300000.0	18500.0	h
136	137	MrBeast2	Vlogs diarios	31300000.0	Estados Unidos	83100000.0	5000000.0	11600.0	

Dataframe shape before and after removing outliers

In [47]:
```
#Before
df.shape
```

Out[47]: (1000, 10)

In [48]:
```
#After
df_no_outliers.shape
```

Out[48]: (849, 9)

Conclusion

In [ ]:
```
The analysis provided valuable insights into top youtube streamers, There were
dataframe, they have been renamed as unknown and Outliers were removed to refi
improve the accuracy of subsequent analysis. The shape of the dataframe was ch
impact of outlier removal.
```

In [ ]:
```

```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: